7. Application Design and Integration

7.1 Application Design

7.1.1 Basic Implementation

Application model. An application consists of the software to perform a set of related tasks. The application is composed of one or more window families, each providing the capabilities to perform one of the tasks, as shown in figure 7-1. The primary window in a family provides control of the overall task, with dialog windows controlling individual subtasks and functions within the task and message windows providing problem, state, or status feedback to users. One task in the application provides application control (i.e., P(AC) in figure 7-1); the primary window for that task opens when the application is started. Navigation to other tasks (i.e., access to other primary windows) in the application may be available only from this window or may be provided in the other primary windows as well.

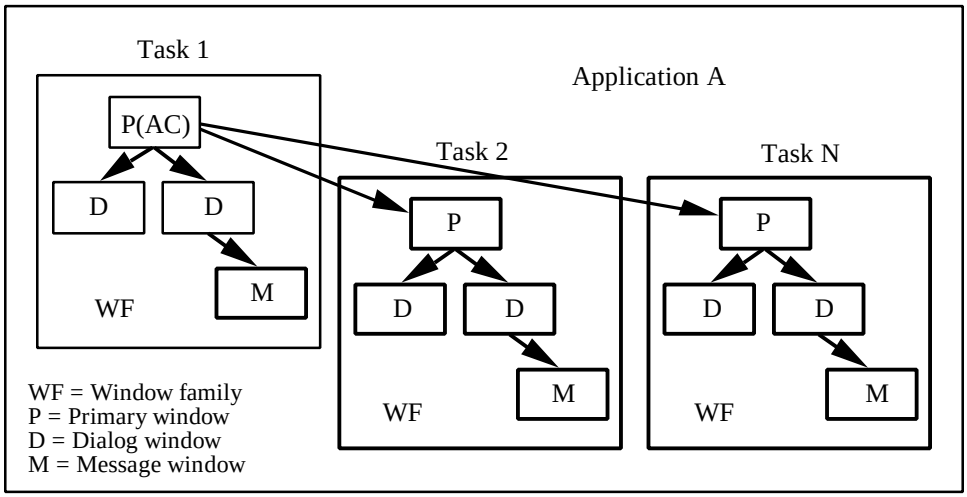

Figure 7-1. Basic implementation of an application.

Starting the application. Users double click on the application icon on the desktop to start the application and open the window with application control. If the application creates document or data files that are represented by a file icon on the desktop, users can also double click on the file icon to start the application and open the file.

Ending the application. Primary windows with a menu bar include a Close option in the File (or comparable) menu that closes it and all of its child windows. Primary windows without a menu bar include a Close push button that performs this action. The window with application control has an Exit option in the File (or comparable) menu that closes all application windows and ends processing by the application.

7.1.2 Variations in Implementation

7.1.2.1 Applications Composed of Nested Segments

Application model. The basic implementation of an application assumes that the software for each task comes from a single source (i.e., one segment). In the DII COE, the software to perform the tasks in an application can be taken from different sources. The result is a nested implementation where the window families in the application may be "owned" by different segments, as shown in figure 7-2. The application is composed of a parent segment that provides application control and one or more child segments that are accessed from within the parent segment. For example, an application might consist of a parent segment that displays the current tactical picture in its primary window, and one or more child segments that are available as map accessories (e.g., function as tactical decision aids). The application icon is named for the parent segment; a primary window in this segment provides application control and opens when the application is started.

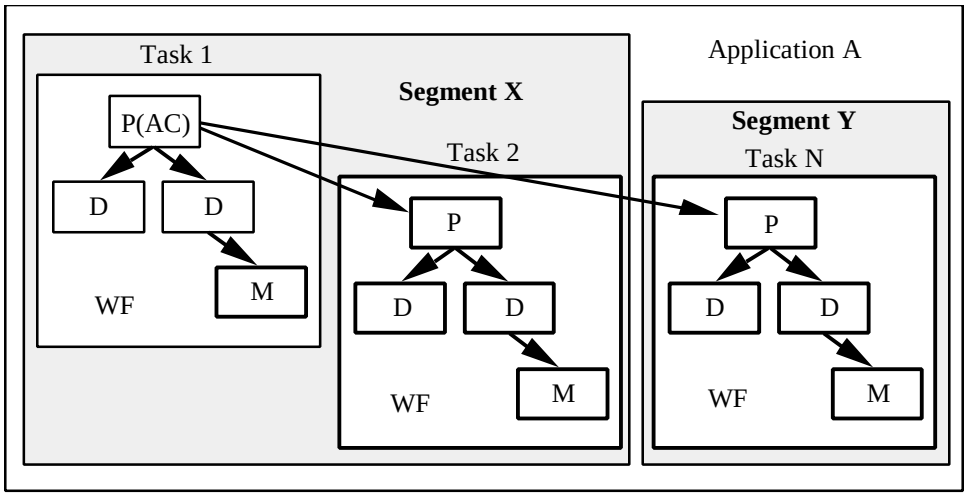

Figure 7-2. Nested implementation of an application.

Accessing segments in an application. In a nested implementation, users access the tasks performed by the child segments from the primary window(s) in the parent segment. The tasks can occupy all or part of a menu in a primary window, be available in a submenu, or be distributed across several menus, as shown in figure 7-3. If menus become excessively long (e.g., extend beyond the bottom of the screen), the application uses submenus, following guidelines in section 5.5, to reduce menu length. Menus that cannot be shortened include controls (e.g., arrow buttons) for scrolling the options that cannot be viewed. Alternatively, the application can reduce menu length by including an option (e.g., in the View menu) for users to show/hide sets of options related to the specific jobs they perform. The parent segment determines when users can and cannot access each of the tasks performed by a child segment and dims the associated menu option(s) to indicate unavailability.

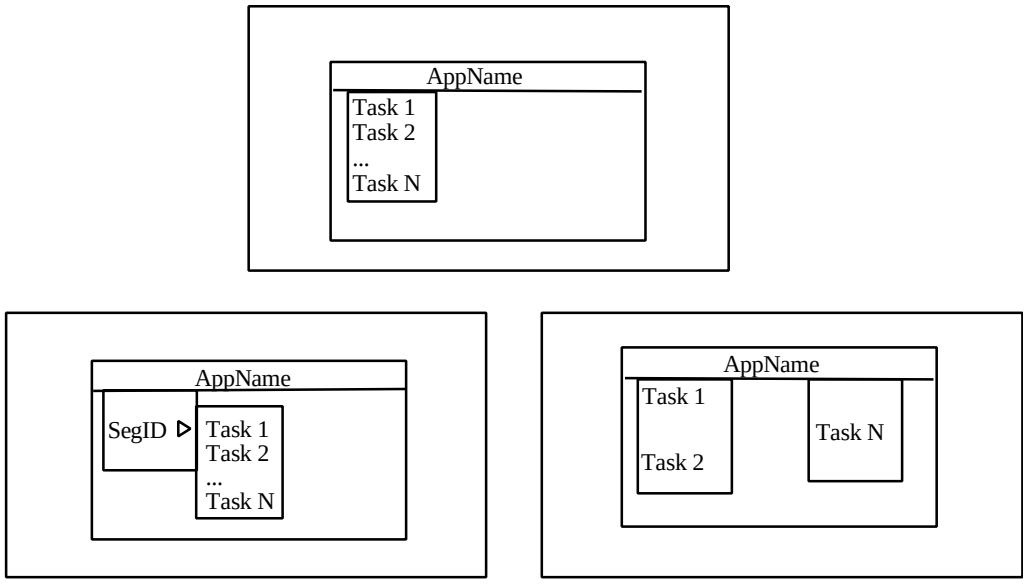

Figure 7-3. Options for accessing child segments in an application.

Starting the application. Users double click on the application icon on the desktop to start the parent segment for the application and open the segment window with application control. Child segments can be started only from within the parent segment.

Ending the application. Closing a primary window (e.g., from a Close menu option or push button, as in the basic implementation) in a parent or child segment closes all windows in the family parented by the window. Exiting the window with application control in the parent segment closes all windows in the application and ends processing in both parent and child segments. An Exit option can be included in the primary window of a child segment, or it can rely on the parent segment to provide this option.

7.1.2.2 Independent and Shared Access to Applications

The COE components in the DII provide the generic support services for performing common functions (e.g., mapping, communications, data management) that may be needed by a mission application. Support services can be available to users from the desktop or accessed from within one or more mission applications either independently or as a shared resource. For example, Application A provides basic map display functions, and Applications X and Y are mission planning tools that generate data for plotting on a map. The map functions in Application A could be accessed as a "task" from within Applications X and Y. These applications could use the functions independently by opening separate map windows to plot their data (i.e., P(1) and P(2) in figure 7-4a), or they could share resources by plotting the data in a single map window (figure 7-4b).

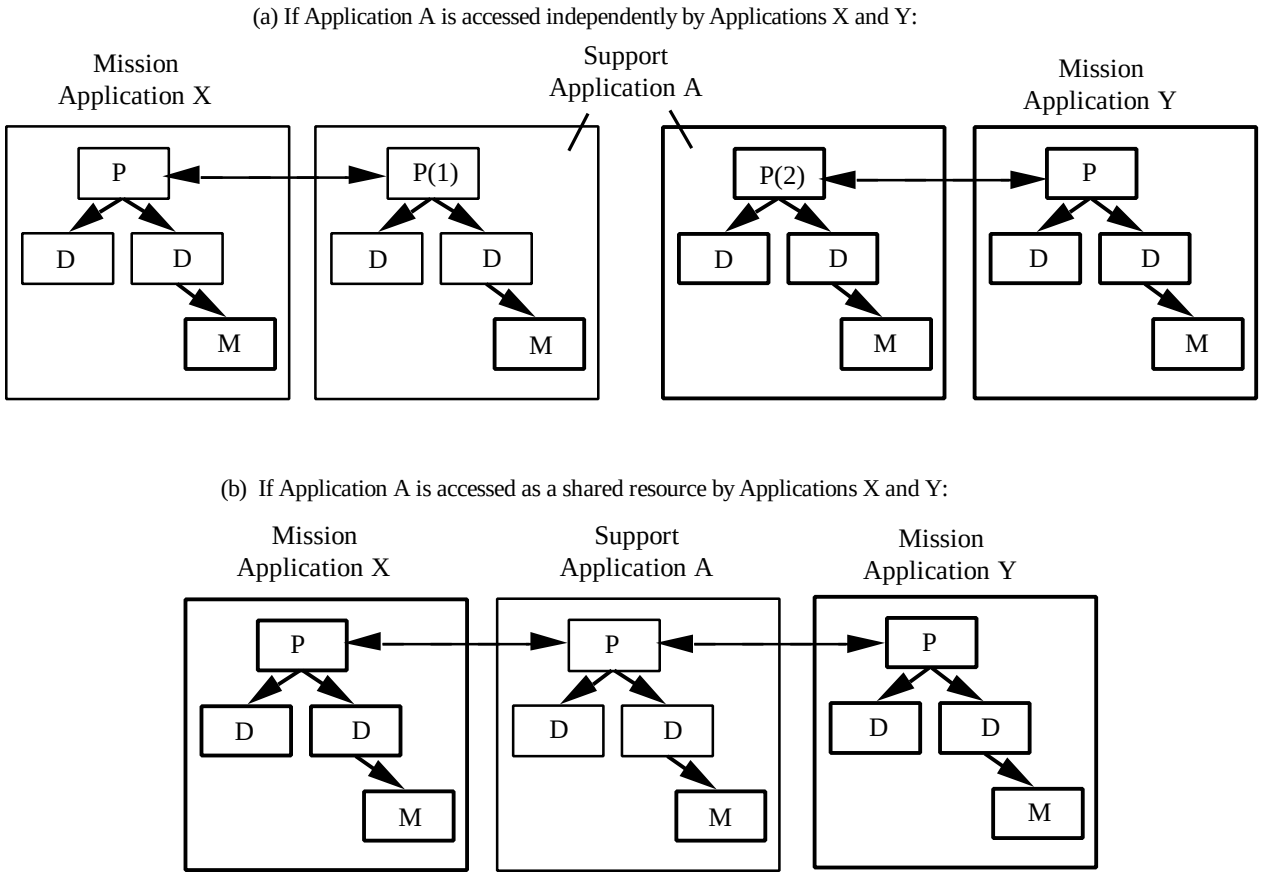

Figure 7-4. Providing independent or shared access to a support application.

Users can access a support application from the desktop (i.e., by double clicking on its application icon) and/or from within one or more mission applications (e.g., from a menu option in a primary window). When the support application is opened as a shared resource and then one of the mission applications is exited, the shared window remains open but the data from the application being exited are removed from the window. When a support application is started from within a mission application but is not shared, the mission application can provide an Exit All option to exit both applications. When a support application is started as a shared resource, the mission application does not include an Exit All option (since the shared window may contain data from an application other than the one being exited).

7.1.2.3 MDI Applications (MS Windows Only)

Application model. When users open a document or data file created by an application, the contents of the file are loaded into the application window that is opened. In a single-document interface (SDI), an application can display only one file in the window at any given time. In a multiple-document interface (MDI), the application can open multiple files simultaneously. Each file is presented in a separate document window that is constrained to appear within a parent application window. The secondary windows in an MDI application are not restricted in this way and can be displayed as in an SDI application. If a menu bar is included in the parent window, its

functionality is shared with the document windows in the application. These windows can also share other interface components (e.g., toolbar, status bar) that are displayed in the parent window.

Starting the application. Users double click on the application icon on the desktop to start the application and open the parent application window. If the application is not running and users double click on a document icon, the parent window opens and the document window is displayed within it. If the application is running and the parent window is already open when users double click on the document icon, a second instance of the parent window opens (with the document window displayed within it), rather than opening the document window in the existing parent window.

Ending the application. Document windows include a Close option (if it has a menu bar) or push button (if it does not have a menu bar) that closes it and all of its child windows. The parent application window includes a Close option or push button that closes it and all of its child document windows. The parent window also includes an Exit option or push button that closes the parent and all child windows and ends processing by the application.

7.1.2.4 Application Groups

A container icon (e.g., a folder) defining an application group can be used to bundle other files (e.g., Read Me text, templates, sample data files) with the application or to provide access to a set of related applications (e.g., a Microsoft Office folder containing Word, Excel, and Powerpoint) on the desktop. Opening the folder displays individual icons for the applications and file in the group. If desired, each application can provide access to the other applications and files in the group by including an Applications menu in the menu bar. Each application in the group is closed and exited independently; if desired, an Exit All option can be included in each application to exit all applications in the group.

7.2 Integrating Applications Within a System

7.2.1 Session Management

A DII system requires users to enter a valid identification and password before a desktop session is initiated and they are able to access any applications. If an application requires additional controls beyond this login procedure, it can provide a separate user login.

If the system provides an indication of classification level on the screen, it uses the classification bar provided by the DII. The default implementation is to display the current classification level in the middle of the bar, with the other parts available to present various status indicators (e.g., alerts) and a digital date-time clock.[1] Appendix E lists the colors to be used when displaying classification level. All classification terms are presented in upper-case letters and spelled out, with caveats abbreviated in accordance with relevant security manuals and directives and with no embedded spaces within words in the label. If the system defines the classification bar as a

[1] An appendix of the DoD style guide provides style guidelines for the Compartmented Mode Workstation.

restricted area of the screen, application windows cannot be placed so that they obscure the classification level.

MS Windows Only: The taskbar is also a restricted area of the screen. Application windows cannot be placed so that they obscure the taskbar.

7.2.2 Availability of Applications on the Desktop

Each application available on the desktop is represented by its own icon (see appendix E). The desktop contains only those applications to which users are granted access; applications (and any associated files or folders) that users are not permitted to access are not visible anywhere on the desktop. Each application determines the constraints for movement and deletion of the data and objects it creates, with users allowed to perform these actions if they have permission to do so. An application cannot affect the availability of other applications on the desktop.

Motif Only: CDE allows users to have several workspaces active on the desktop. When users open a new application window, it is displayed in the current workspace and only occupies that workspace. When users move an application window between workspaces, it and all of the other windows in its window family move together.

The system may allow users to rename an application (i.e., change the label in the application icon) and remove it from the desktop (e.g., drag the application icon to the trash), or may restrict users from performing these actions. The system may also allow users to create new container icons (e.g., folders) within which to place individual applications. If users are allowed to start only one instance of an application, double clicking on the icon for an application that is running only raises any open window(s) for the application to the top of the window stack. If multiple instances of an application can be running, users double click on the application icon each time they want to start another instance of the application.

7.2.3 Style Management

The desktop in CDE and MS Windows allows users to configure various style-related features that are applied to all of the applications in the system. As a result, individual applications need to be designed so they can respond to any changes in appearance and behavior defined at a system level. This section addresses some of the style features that can be customized in systems with a CDE or MS Windows desktop.

Color. Both CDE and MS Windows allow users to select from a set of predefined color palettes (or schemes), modify these palettes, and create custom palettes. It is expected that DII systems will identify a set of palettes that are appropriate to the operational environment in which the system will be used and will configure the color manager provided by the desktop to contain this set. Systems are expected to implement the centralized color management capability provided by the desktop so that applications change color dynamically when users select a different color palette. An application that cannot use dynamic colors implements the default DII palette as its color set. This palette provides a light gray background with black text, assuming an office-like operational environment with normal ambient lighting. In addition, because users can choose

among various color palettes, the application is designed so that any application-specific color coding is legible regardless of the palette selected. Appendix E provides additional information on color implementation in Motif and MS Windows.

Font. CDE uses a proportional-width font to display system text (e.g., in menu bars, push buttons, labels) and a fixed-width font to display text entered by users. The CDE style manager provides users with a choice of seven sizes for displaying the system and user fonts. The MS Windows style manager allows users to select the name, size, and color of the font used to display text in individual window components. As with color, it is expected that DII systems will implement the centralized font management capability provided by the desktop. If the application cannot change font dynamically, it uses the default fonts defined by Motif and MS Windows (see section 12.1.1 and appendix E). Because users can choose among various font attributes, the application is designed so that all of the text it displays adapts correctly when these attributes are changed (e.g., text is not cropped, does not wrap inappropriately). Appendix E provides additional information on font implementation in Motif and MS Windows.

Input focus. As indicated in section 3.1, users can assign input focus to a window either explicitly or implicitly. The style manager in CDE allows users to select between these focus modes. While explicit focus is considered the default by DII, systems may choose to implement implicit focus or provide users with the flexibility to select either focus policy. As a result, the application is designed so that it can support both explicit and implicit focus (i.e., ensure that application-unique features designed for use with implicit focus also behave correctly if explicit focus is selected, and vice versa). MS Windows implements explicit focus and does not provide a choice of focus modes as a style feature that can be configured at a system level. While it is possible for individual applications to manage window focus implicitly, DII specifications do not support this implementation because it provides an inconsistent interaction model when the application is integrated with other applications that rely on explicit focus.

Pointing device features. The style manager in CDE and MS Windows allows users to reverse the actions assigned to BLeft and BRight (so that the pointing device can accommodate differences in handedness), to set the double-click speed, and to adjust the speed with which the pointer moves across the display. In CDE, users can also set whether BMiddle is used to extend selections or perform drag transfer actions, and in MS Windows, users can change the shape of the pointer and enable pointer “warping” that automatically places the pointer to the default push button in a dialog box. Because users can vary these attributes of the pointing device, the application is designed so that any application-specific pointer shapes or pointer actions do not conflict with pointer settings defined at a system level.

This page intentionally left blank.